Looping through, we reach the first node 1, which contains the edge list of 3, 2. Using the algorithm, we see 3 hasn't been added so we keep it. Then check 2, which we keep and see it is smaller than 3 so we put that to the front of the list, giving us $1 \rightarrow 2\ 3$

Next, we have node 2 with $2 \rightarrow 1, 3, 3, 3$. Repeating this process, we check 1 and keep it, check 3 and keep it which is greater than 1 so it stays in the same position. The next 2 are already contained in the list, so we just delete the representation giving us $2 \rightarrow 1\ 3$

We repeat this for the rest of the adjacency list and we come to our (ii) case at node $4 \rightarrow 4, 5, 2, 5, 1$. Since we want to remove self-loops, node 4 cannot have an edge with itself, so its deleted and the process continues from above giving us $4 \rightarrow 1, 2, 5$.

Thus, the algorithm will convert G into G' removing self loops and making a directed multiedge into a single directed edge.

Pseudocode input is the adjacency list.

```
transform(G)
  loop through G //get each vertex linked-list
    loop through G[i] linked list
      if(G[i] = edge[i])
        delete edge[i] //removes self loop
      else if (edge[i] = visited)//already in list
          delete edge[i]
      else
          if(edge[i] < edge[i-1]
            swap edge[i] and edge[i-1]
          edge[i]=visited
          add edge[i] to linked list
          i++
```